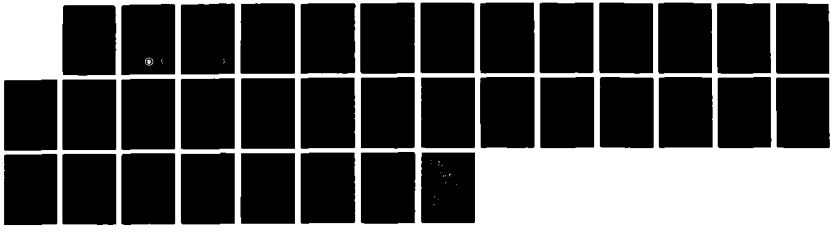
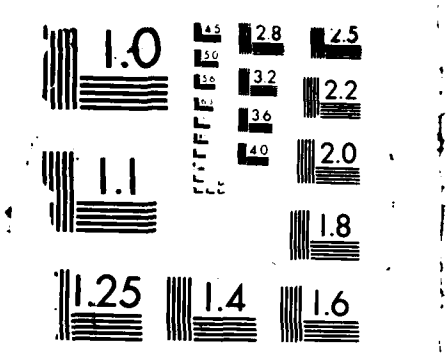
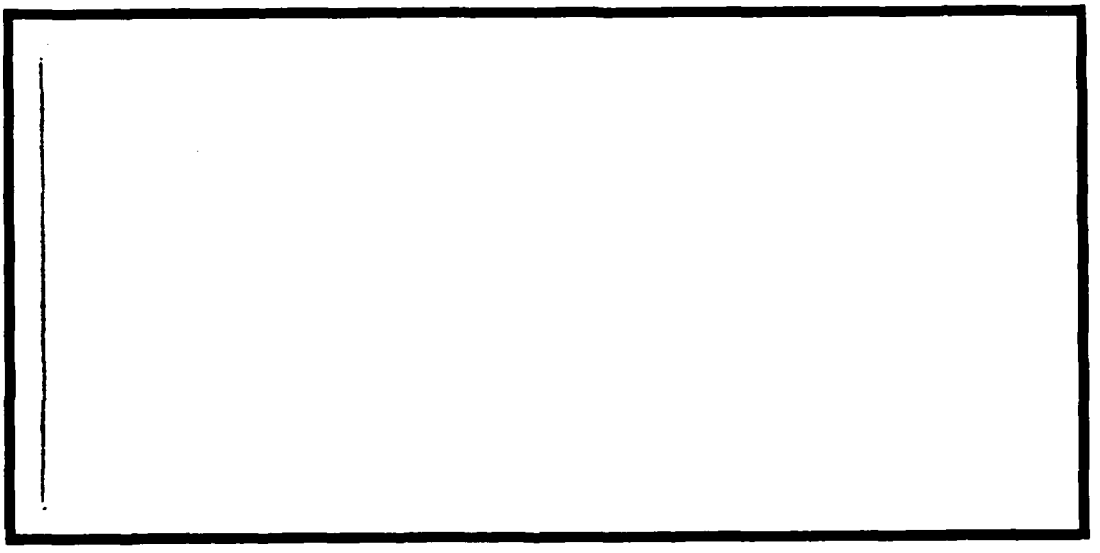
AD-A186 559 MULTICOLOR NUMBERING OF IRREGULAR TRIANGULAR MESHES(U) 1/1
PITTSBURGH UNIV PA INST FOR COMPUTATIONAL MATHEMATICS
AND APPLICATIONS R G MELHEM ET AL. JUN 87 ICMA-87-110
UNCLASSIFIED N00014-85-K-0339 F/G 12/3 NL





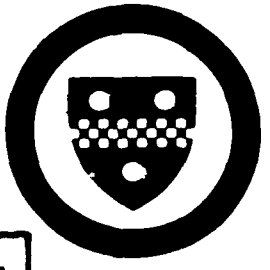
(14)

INSTITUTE FOR COMPUTATIONAL MATHEMATICS AND APPLICATIONS



Department of Mathematics and Statistics

University of Pittsburgh



(14)

Technical Report ICMA-87-110 June 1987

Multicolor Numbering of Irregular Triangular Meshes[1]

by

Rami G. Melhem[2] and K.V.S. Ramarao[3]

1. This work, in part, is supported under ONR contract N00014-85-K-0339 and Air Force contract AFOSR-84-0131.

2. Department of Computer Science and Institute of Computational Mathematics and Applications, Department of Mathematics and Statistics, University of Pittsburgh, Pittsburgh, PA 15260

3. Department of Computer Science, University of Pittsburgh, Pittsburgh, PA 15260

MULTICOLOR NUMBERING OF IRREGULAR TRIANGULAR MESHES

Rami G. Melhem

and

K.V.S. Ramarao

Dept. of Computer Science

The University of Pittsburgh

Pittsburgh, PA 15260.

ABSTRACT

Many iterative algorithms for the solution of large linear systems may be effectively vectorized only if the diagonal of the matrix is surrounded by a large band of zeroes, which is called the zero stretch. In this paper, a multicolor numbering technique is suggested for maximizing the zero stretch of irregularly sparse matrices. The technique, which is a generalization of a known algorithm for regularly sparse matrices, executes in linear time, and produces a zero stretch approximately equal to $n/2\sigma$, where σ is the number of colors used in the algorithm. For triangular meshes, it is shown that $\sigma \leq 3$, and that it is possible to obtain $\sigma=2$ by applying a simple backtracking scheme.

This work is, in part, supported under ONR contract N00014-85-K-0339, and the Air Force contract AFOSR-84-0131.

1. INTRODUCTION

Iterative solution techniques, in general, and preconditioned conjugate gradient methods (PCCG), in particular, are becoming more and more attractive for the solution of sparse linear systems. Namely because these methods preserve sparsity, and thus allow for the solution of large systems without resorting to auxiliary (off core) storage. This is a big advantage over direct solution methods that destroy sparsity. Moreover, the availability of supercomputers contributed to the popularity of the iterative techniques by providing the computational power necessary for the solution of huge linear systems, and yet not providing enough memory for storing the systems in their dense (or banded) forms.

However, the power of supercomputers may be fully exploited only for computations which are vectorizable. Unfortunately, one of the best known iterative techniques for the solution of symmetric, positive definite, linear systems, namely the incomplete factorization/PCCG method [11, 13], requires the solutions of triangular linear systems of equations. The algorithms for the solution of such systems are highly recursive, and thus poorly vectorizable. More specifically, for the solution of a linear system $Ax = b$, each PCCG iteration requires the solution of two triangular systems of the forms $(L+D)x = b$ and $(L+D)^T x = b$, where L is a strictly lower triangular matrix and D is a diagonal matrix. Both L and D are derived from A, and, for irregularly sparse systems, the sparsity structure of $L+D+L^T$ is identical to the sparsity structure of A.

The recursive solution algorithms for $(L+D)x = b$ and $(L+D)^T x = b$ are only vectorizable to the extent permitted by the structure of L. More specifically, these algorithms may be written in terms of vectors of length Δ, where Δ is the size of the maximum band which contains zeroes below the diagonal of L (for more details see [15]). A more precise definition of Δ may be given as follows

Definition: The zero stretch of an $n \times n$ strictly lower triangular matrix L is the largest integer Δ such that $a_{i,j} = 0$ for $i = 1,\ldots,n$ and $i < j \leq i - \Delta$. In this paper, we assume that $L+D+L^T$ has the same sparsity structure as A, and we define the zero stretch of A

to be equal to the zero stretch of L.

Hence, for the efficient solution of $(L+D)x=b$ and $(L+D)^T x=b$ on vector supercomputers, it is essential to rearrange the rows and columns of the $n \times n$ symmetric matrix A with the goal of maximizing the zero stretch. In order to achieve this goal, we consider the graph G_A corresponding to A. More precisely, G_A contains n nodes which are numbered by the integers $1, \cdots, n$ such that the node numbered i corresponds to row/column i in A, and G_A contains an edge from node i to node j if and only if the $(i,j)^{th}$ element of A is non-zero. Now, interchanging rows/columns i and j in A is equivalent to interchanging the numbers of the two corresponding nodes in G_A. In other words, maximization of the zero stretch of A may be obtained by renumbering the nodes of G_A. That is by assigning to each node i, $1 \leq i \leq n$, a unique number $\nu(i)$, $1 \leq \nu(i) \leq n$, such that the difference between $\nu(i)$ and $\nu(j)$ for any two neighboring nodes i and j in G_A is as large as possible.

Clearly, the problem of maximizing the zero stretch of a matrix is, in some sense, the reciprocal of the problem of minimizing the bandwidth of the matrix [18]. The latter problem is one of minimizing the difference $|\nu(i)-\nu(j)|$, and is important for direct solutions of linear systems. Both problems have been proven to be NP-hard (see [5, 10, 16]). However, many heuristics have been studied for bandwidth minimization (see for examples [4, 6, 12, 21]), while, to our knowledge, the problem of zero stretch maximization have been studied only in the context of matrices resulting from the discretization of partial differential equations (PDE) on regular grids (see for example [1, 7, 8, 14, 17, 19]).

If a matrix A results from the discretization of a PDE on a regular grid, then its corresponding graph, G_A, has a repetitive structure which allows for a simple numbering scheme. Very briefly, a systematic way may be used to color the nodes in G_A such that any two adjacent nodes have different colors. Then, consecutive numbers are assigned to the nodes that have the same color. If n_c is the number of nodes assigned the color c and $n_{min} = \min_c \{n_c\}$, then the zero stretch resulting from the multicolor numbering is approxi-

mately equal to n_{min}.

Multicolor numbering schemes may be generalized and applied to pierced rectangular grids [14, 15] and to irregular grids. In this paper, a multicolor numbering of irregular triangular grids is considered, and its effect on the zero stretch of the corresponding matrices is studied. Both theoretical and experimental results are presented.

The numbering algorithm is described in Section 2. Briefly, the basic step in the algorithm is the construction of a level structure similar to the one used in the Cuthill-McKee algorithm [4]. The levels are, then, grouped into two groups, namely the odd numbered levels and the even numbered levels, and the nodes in each level are colored such that neighboring nodes have different colors. Finally, the nodes which have the same color in the odd numbered levels are numbered consecutively, and the same is repeated for the even numbered levels.

The efficiency of the algorithm is studied in Section 3, and a lower bound on the resulting zero stretch is obtained. This bound shows that larger zero stretches may be obtained by reducing both the number of colors, σ, needed to color each level, and the difference, α_{max}, between the number of nodes in the odd numbered levels and the number of nodes in the even numbered levels.

The theoretical lower bound on the zero stretch motivates two modifications of the basic algorithm. The first modification, which is described in Section 4, aims at the reduction of α_{min}, and the second, which is described in Section 5, aims at the reduction of σ. Finally, in Section 6, the results of running the algorithm on three irregular meshes are presented and discussed.

2. DESCRIPTION OF THE ALGORITHM

The class of meshes considered in this study consists of irregular meshes of triangular elements, where each element contains three nodes located at the corners of the triangle. If a mesh in the above class is used to discretize a PDE, then the resulting matrix A corresponds to a graph G_A which has the same topology as the mesh itself. For this reason, we will not differentiate, from now on, between the mesh and the graph G_A.

Given a graph G_A in which nodes are identified by the integers $1, \cdots, n$, the problem is to find the numbering function, $\nu:[1,n] \rightarrow [1,n]$, which maximizes the following

$$\Delta = \min\{ |\nu(i)-\nu(j)| \; ; i \text{ and } j \textit{ are neighbors in } G_A \}$$

Unfortunately, the number of possible numbering functions is $n!$ (factorial n), and the specific numbering that maximizes Δ may not be found without trying (in the worst case) all of the $n!$ numberings. In the remaining of this section, we will describe an algorithm which executes in linear time and finds a numbering that produces a large Δ. The basic idea of the algorithm is the inclusion of each node of G_A into one of t lists, namely $L^1, \cdots, L^t$, such that:

1) Any two neighboring nodes in G_A are not in the same list. That is, any given list contains independent nodes, where two nodes are called independent if they are not neighbors.

2) Let ϕ be the smallest constant such that if the k^{th} node in some list and the l^{th} node in another list are neighbors in G_A, then $|\, l - k \,| \leqslant \phi$. The order of the nodes in each list should lead to a small ϕ.

Given these lists, the nodes in the first list may be numbered consecutively, followed by the nodes in the second list, then the third , and so on. If the number of nodes in list L^i is denoted by $|L^i|$, then, the above scheme gives

$$\Delta \geqslant \min_{1 \leqslant i \leqslant t} \{ |L^i| \} - \phi \qquad (1)$$

Assuming that $\phi << \min\{|L^i|\}$, then, larger values of Δ may be obtained by maximizing the size of the shortest list. That is, by using the minimum number of lists and making the lists approximately equal in size. These were the major factors that guided the design of our algorithm.

The first step toward the construction of the lists $L^1, \cdots, L^t$ is the generation of a level structure $\{V_1, \cdots, V_m\}$ that covers G_A. This structure is similar to the one used in the Cuthill-Mckee algorithm [4]. More specifically, the generation starts from an initial level V_1 and proceeds inductively such that, given any level V_u, the next level V_{u+1} is generated by including in it any node which is a neighbor to a node in V_u and which have not been included in any of the previous levels $V_1, \cdots, V_u$. The number of nodes in V_u is denoted by $|V_u|$. Note that, by construction, neighboring nodes may lie only on the same level or on adjacent levels. That is, a node in a given level V_u is independent of any node in levels V_w, $w > u+1$ or $w < u-1$.

The nodes in each level V_u are then partitioned into the minimum number of sets $V_u^1, V_u^2, \cdots$, each containing only independent nodes. In other words, no two nodes in the same set V_u^i may be neighbors. In Section 3, it is proved that three sets are always sufficient to accomplish this partition for any level. Hence, if the number of sets required to partition level V_u is denoted by s_u, then, $s_u \leqslant 3$. Let $\sigma = \max_{1 \leqslant u \leqslant m} \{s_u\}$, and for any level u with $s_u < \sigma$, define the sets V_u^i, $i = s_u + 1, \cdots, \sigma$ to be empty.

Now, it can be readily seen that, by picking one set from each alternate level, a list of independent nodes may be compiled. For example, all the nodes in $V_1^1, V_3^1, V_5^1, \cdots$ are independent, and hence, may be included in one list. Using this approach, all the nodes of G_A may be distributed into 2σ lists of independent nodes; σ of these lists may be constructed from the odd numbered levels, and the other σ from the even numbered levels. In Fig 1, a logical representation of the level structure is shown, where two sets of independent nodes are connected by an edge if and only if there exists two nodes, one in each set, that are neighbors in G_A. A list is highlighted in the figure by using the same hashing

direction for all of its sets.

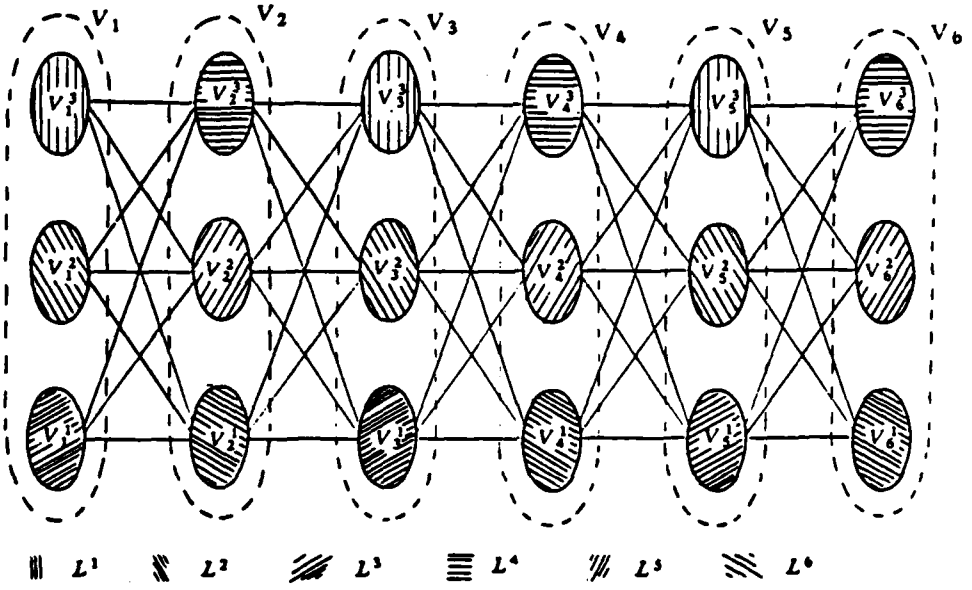


Fig 1 - The partitioning of the level structure (σ=3)

However, the above construction may produce lists that vary considerably in size, with some lists being very small. By (1), this leads to a small Δ. In fact, the largest Δ is obtained if the 2σ lists are of equal size.

It is possible to balance the size of the lists generated from odd (or even) numbered levels. For example, assume that the lists $L^1, \cdots, L^\sigma$ are generated by picking sets from the odd numbered levels. If after picking sets from $V_1, V_3, \cdots, V_{u-2}$, the lists are ordered (and renamed) such that their sizes satisfy $|L^k| \leqslant |L^{k+1}|$, for $k=1, \cdots, \sigma-1$, then, the sizes of the lists may be balanced by first renaming the sets $V_u^1, \cdots, V_u^\sigma$ such that $|V_u^k| \geqslant |V_u^{k+1}|$, and then adding V_u^k into L^k for $k=1, \cdots, \sigma$. If this process is applied at all the odd levels, $u=1,3, \cdots$, then, it is straight forward to prove the following using a recursive argument:

$$\| |L^i| - |L^j| \| \leqslant \max\{|V_u^k| : u = odd, \text{ and } k=1, \cdots, \sigma\} \qquad i, j \leqslant \sigma \qquad (2.a)$$

where the outer most vertical bars in the left side of the inequality are used to indicate the absolute value of a signed integer. Similarly, if the lists $L^{\sigma+1}, \cdots, L^{2\sigma}$ are generated from the even numbered levels, then

$$| \, |L^i| - |L^j| \, | \leq \max\{|V_u^k| \, ; u = even \text{ , and } k = 1, \cdots, \sigma\} \qquad i, j \geq \sigma+1 \text{ (2.b)}$$

With the above strategy, the size of the lists $L^1, \cdots, L^\sigma$, as well as the size of the lists $L^{\sigma+1}, \cdots, L^{2\sigma}$ may be balanced. However, the size of a list L^i, $i \leq \sigma$ relative to another list L^j, $j > \sigma$, depends on the total number of nodes in the odd and even numbered levels. That is, $|L^i| \approx |L^j|$ only if $\sum_{u=odd} |V_u| \approx \sum_{u=even} |V_u|$. For the examples considered in Section 6, these sums were approximately equal. However, should there be a big difference between $\sum_{u=odd} |V_u|$ and $\sum_{u=even} |V_u|$, then the 3-way grouping algorithm described in Section 4 may be used.

In brief, the multicolor node numbering scheme may be summerized as follows:

NODE NUMBERING ALGORITHM

Initially, input the nodes in level V_1, and set $u = 1$

1) partition V_1 into the sets $V_1^1, \cdots, V_1^{s_1}$ such that each set contains independent nodes, and $|V_1^k| \geq |V_1^{k+1}|$, $k = 1, \cdots, s_1 - 1$.

2) REPEAT until every node is in some level

   2.1) $u = u + 1$

   2.2) Construct the next level V_u

   2.3) Partition V_u into the sets $V_u^1, \cdots, V_u^{s_u}$ such that the nodes in each set are independent, and $|V_u^k| \geq |V_u^{k+1}|$, $k = 1, \cdots, s_u - 1$.

3) Let m be the number of levels obtained in step 2. and set $\sigma = \max\{s_u \, : u = 1, \cdots, m\}$.

4) FOR $i = 1, \cdots, 2\sigma$, let L^i = the empty list.

   FOR $u = 1, \cdots, m$, with an increment of 2, DO

   4.1) Find two permutations; π on $[1, \sigma]$ and ρ on $[\sigma+1, 2\sigma]$, such that

$$|L^{\pi(k)}| \leq |L^{\pi(k+1)}|, \qquad k = 1, \cdots, \sigma-1.$$
$$|L^{\rho(k)}| \leq |L^{\rho(k+1)}|, \qquad k = \sigma+1, \cdots, 2\sigma-1.$$

4.2) FOR $k = 1, \cdots, \sigma$, append the nodes in V_u^k to the list $L^{\pi(k)}$.

4.3) FOR $k = \sigma+1, \cdots, 2\sigma$, append the nodes in $V_{u+1}^{k-\sigma}$ to the list $L^{\rho(k)}$.

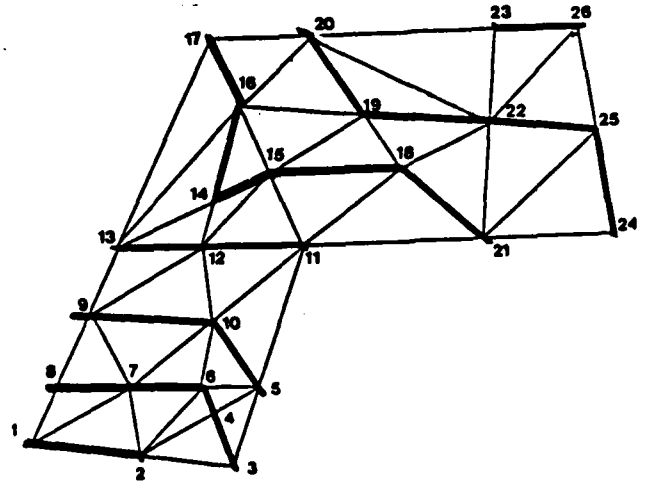
5) $\lambda = 0$

FOR $i = 1, \cdots, 2\sigma$ DO

FOR each node e in the list L^i DO

$\lambda = \lambda + 1 \quad ; \quad \nu(e) = \lambda$

It is easy to see that the execution time of the above algorithm is proportional to the number of edges in G_A. More specifically, the most time consuming part in the algorithm is the construction of the level structure, which depends on the number of edges in the graph.

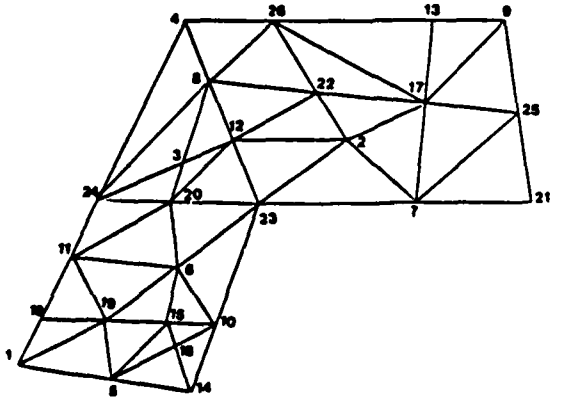


(a) an arbitrary numbering

V_u^k	$k=1$	$k=2$	$k=3$
$u=1$	1	2	
$u=2$	3,6,8	4,7	
$u=3$	5,9	10	
$u=4$	11,13	12	
$u=5$	18,14,17	21,16	15
$u=6$	25,20	24,19	22
$u=7$	23	26	

$L^1 = (1,18,14,17)$
$L^2 = (2,10,21,16,26)$
$L^3 = (5,9,15,23)$
$L^4 = (3,6,8,22)$
$L^5 = (4,7,12,24,19)$
$L^6 = (11,13,25,20)$

(b) the construction of the lists



(c) The multicolor numbering

Fig 2 - The numbering of an irregular mesh.

In Fig 2, the algorithm is applied to a simple mesh which consists of 26 nodes. The levels are depicted by bold lines, and the sets of independent nodes within each level, as well as the lists of independent nodes are identified in Fig 2(b) in terms of the arbitrary numbering used in Fig 2(a). The numbering resulting from applying the algorithm is shown in Fig 2(c), from which it may be seen that $\Delta = 3$. If {24,25,26} is chosen as a starting level, or if the backtracking algorithm described in Section 4 is applied, then the value of σ may be reduced from 3 to 2, and $\Delta = 4$ is obtained. For this small problem, it was possible to find the numbering which maximizes Δ by doing an exhaustive search which took 382 hours to execute on a SUN3 workstation (a problem with 28 nodes would need $27 \times 28 \times 382$ hours). The largest Δ was found to be equal to 7 for this mesh.

In the next section, a lower bound on the value of the zero stretch, Δ, resulting from the algorithm is established.

3. ANALYSIS OF THE ALGORITHM

The problem of partitioning the nodes in a level into sets of independent nodes is equivalent to the problem of coloring the nodes in the level such that any two neighboring nodes have different colors. Given such a coloring, the sets may, then, be formed by including nodes with the same color in the same set. Hence, from now on, if a minimum of s independent sets are required to partition a specific level, then we will say that this level is s -colorable.

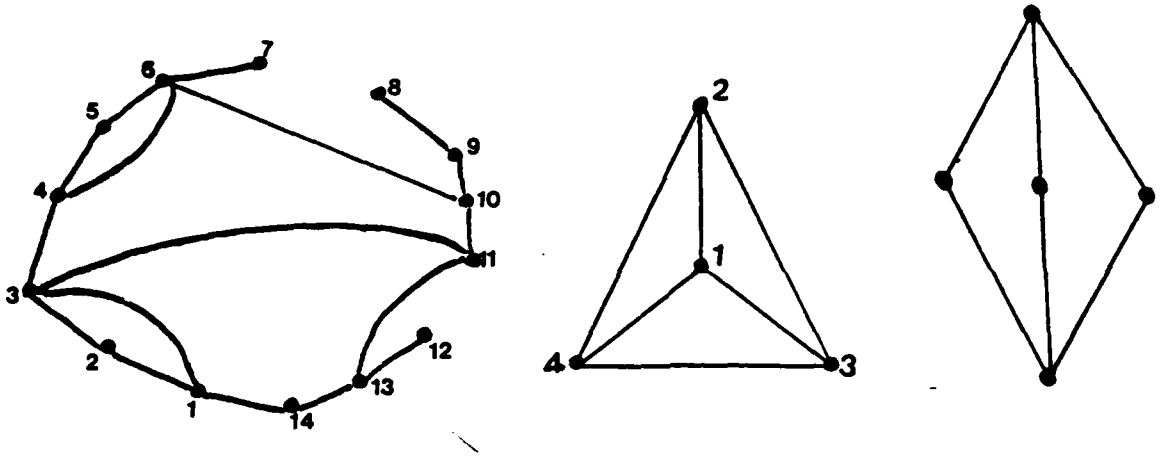
In this section we analyze the performance of the algorithm and obtain a lower bound on Δ produced by the algorithm. We need to digress slightly in doing so and study certain properties of planar and outerplanar graphs. Specifically, we proceed as follows : first we observe that G_A is a planar graph; next we show that when the algorithm is executed on a planar graph, if the initial level taken by the algorithm forms an outerplanar graph, then each of the subsequent levels also forms an outerplanar graph; then, we show that any outerplanar graph can be colored using no more than three colors; finally this result is used to obtain lower and upper bounds on the performance of the algorithm.

3.1. Outerplanar graphs

Formally, [2, 3, 20], an outerplanar graph is defined as follows:

Definition: A graph is said to be planar if there exists a mapping of its vertices and edges into the plane such that a) each vertex v is mapped into a distinct point v' in the plane, b) each edge (v,w) is mapped onto a simple curve having end points (v',w'), so that c) the mappings of the edges meet only at common end points. A planar graph is said to be outerplanar if it can be embedded in the plane such that all vertices lie on some contour and all the edges lie inside this contour (see Fig 3(a)).[]

Definition: A subdivision of a graph G is a graph G', obtained from G by replacing some edge (u,v) by a new vertex w together with the edges (u,w) and (w,v). A graph H is homeomorphic from a graph G if H can be obtained from G by a finite sequence of subdi-



(a) An outerplanar graph (b) A K_4 graph (c) A $K_{2,3}$ graph

Fig 3 - Some planer graphs

visions. []

The following result is known previously [2]:

Theorem: Let K_4 denote the complete graph on 4 vertices and $K_{2,3}$ the complete bipartite graph on 5 vertices (see Fig 3). Then a graph G is outerplanar if and only if G has no subgraph homeomorphic from K_4 or $K_{2,3}$. []

3.2. An upper bound on σ

In order to determine the maximum number of colors needed to color the nodes in any particular level, we start by proving the following:

Proposition: Let $G=(V,E)$ be a planar graph and let $S=(V',E')$ be a connected subgraph of G. If $S=V_k$ for some k during the execution of the node numbering algorithm of Section 2, then V_{k+1} is outerplanar.

Proof: Assume, for contradiction, that V_{k+1} is not outerplanar. Thus it must have a subgraph homeomorphic from K_4 or $K_{2,3}$. Since we are investigating meshes with triangular elements only, we observe that if G_A contains a subdivision of K_4 or $K_{2,3}$, then it contains K_4 or $K_{2,3}$ respectively. Assume that K_4 is a subgraph of V_{k+1} and number its vertices 1,2,3,4 arbitrarily. Since S is connected and G_A is planar, S must be entirely contained either in one of the internal regions (1,4,3), (1,2,4), and (2,3,4) or must be outside the K_4 (that is, no vertex of S is in any of these regions) in any planar embedding of G_A. It is clear

that whatever be the case, at least one edge connecting a vertex of S to one of the vertices of K_4 must necessarily cross one of the edges of K_4. Similarly we can show in the case of $K_{2,3}$ that at least one vertex of $K_{2,3}$ cannot be connected to any vertex of S without crossing an edge in any planar embedding of G_A.[]

In the statement of the above proposition, it is necessary that S be connected. Counter examples can be easily constructed with S spread across several regions of K_4 or $K_{2,3}$ while still guaranteeing the planarity of G_A. The proposition implies that if the initial level chosen by the algorithm is connected, then each of the levels generated later is an outerplanar graph. This is true even if the levels other than the initial level are not connected since the planarity of G_A ensures that no level generated by the algorithm is spread across the regions of a K_4 or $K_{2,3}$. Note that the connectedness of the initial level is a sufficient and not a necessary condition. In other words, it is possible, for some meshes to start with an initial level that is not connected and yet generate only outerplanar graphs as the subsequent levels.

The following result concerning the 3-colorability of outerplanar graphs was known previously [2, 3, 20]. However, we have not been able to find an explicit algorithm in the literature which actually provides the coloring. We present such an algorithm here, which is actually programmed as part of the experiments reported in Section 6.

Theorem: Any outerplanar graph is 3-colorable.

Proof: We consider outerplanar graphs which do not have isolated nodes. The coloring of isolated nodes is trivial becauser they do not have any neighbor. Given an outerplanar graph G with N nodes, we first line up its nodes on a straight line such that all the edges are on one side of the line. This is always possible by chosing any two adjacent nodes on the contour of the planer embedding of the outerplanar graph, and considering these two nodes to be the first and the last node in the node lineup. All other nodes may be lined up in the order of their appearance on the contour (see Fig 4). In order to identify the nodes, we number them sequentially starting at one of the end nodes.

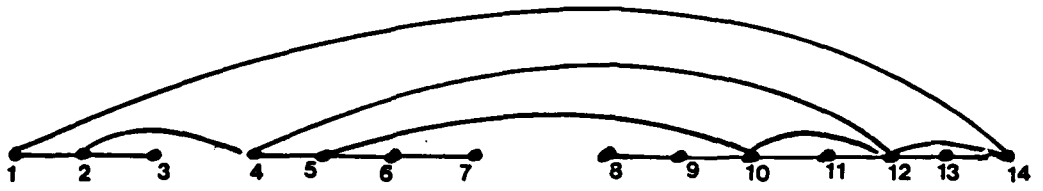


Fig 4 - A different embedding (a line up) of the graph of Fig 3(a)

For any two neighboring nodes with numbers v and w (say $v < w$), we define the region $r(v,w)$ to be the set of nodes with numbers between v and w, including v and w. Note that if $w = v+1$, then $r(v,w)$ consists of only w and v. The following statement follows from the definition of outerplanarity: if a vertex u, different from v and w, is in $r(v,w)$, then any region $r(u,e)$, which has u as one of its end nodes, should be properly contained in $r(v,w)$. The nesting level of a region is the number of regions which properly contain that region. For example, a region $r(v,w)$ is called 0-nested if it is not contained in any other region. That is, if there do no exist two nodes $e_1 \leq v$ and $e_2 \geq w$, such that e_1 and e_2 are neighbors, and at most one of e_1 or e_2 is equal to v or w, respectively.

The coloring strategy is recursive with respect to the nesting level. More specifically, assume that $r(u_1,w_1)$, $r(u_2,w_2)$, $\cdots$, $r(u_k,w_k)$, $k \geq 1$, are the 0-nested regions in G. Noting that $u_1=1$ and $w_k = N$, we define the set $R_0(1,N)$ to contain all the end nodes of the 0-nested regions between nodes 1 and N. That is $R_0(1,N) = \{u_i, w_i \mid i=1,\ldots,k\}$. Clearly, for any two consecutive regions, either $v_i = w_{i+1}$ or $v_i = w_{i+1} - 1$. In the former case, node v_i has two neighbors among the nodes in $R_0(1,N)$, and in the latter case, v_i has only one neighbor among the nodes in $R_0(1,N)$. This argument applies to any node in $R_0(1,N)$, and hence, it is possible to color all the nodes in $R_0(1,N)$ using only three colors (call these colors c 1, c 2 and c 3). In fact, this last statement is true even if the colors of the first and last nodes, namely 1 and N, are bound a priori to two of the colors c 1, c 2 and c 3.

After coloring the nodes in $R_0(1,N)$, we consider the 0-nested regions consecutively. Each 0-nested region $r(u_i,w_i)$ is either a trivial region which contains only nodes u_i and

w_i, or it contains at least two 1-nested regions $r(u_{i,1},w_{i,1}), \cdots, r(u_{i,p},w_{i,p})$, where, $u_{i,1}=u_i$, $w_{i,p}=w_i$ and $p \geq 2$. Given that u_i and w_i are already colored with two different colors from $\{c\,1, c\,2, c\,3\}$, then nothing has to be done in the former case. If, however, the latter case applies, then we consider the set $R_1(u_i,w_i)$ which contains the end nodes $u_{i,j}$ and $w_{i,j}$, $j=1,\ldots,p$, of the 1-nested regions between u_i and w_i. Following an argument similar to the one used for $R_0(1,N)$, it may be shown that the nodes in $R_1(u_i,w_i)$ may be colored using only $c\,1$, $c\,2$ and $c\,3$.

After coloring the nodes in $R_1(u_i,w_i)$, we repeat the above process with the 2-nested regions contained in each 1-nested region $r(u_{i,j},w_{i,j})$, $i=1,\ldots,p$. When the recursive process terminates, all the nodes in G will be colored. Termination is guaranteed because every region in G has a finite nesting level. []

Corollary: Each of the levels generated by the node numbering algorithm of Section 2. is 3-colorable. []

3.3. A lower bound on Δ

In this section, Equation (1) is used to derive a lower bound on Δ in terms of the total number of nodes n in the graph G_A. First, some terminology is introduced: Let $S_{even,u} = \{V_i : i = even \text{ and } i \leq u\}$ be the set that contains the even numbered levels, up to level u, and let $n_{even,u}$ be the total number of nodes in these levels, that is $n_{even,u} = \sum_{V_i \in S_{even,u}} |V_i|$. Let $S_{odd,u} = \{V_i : i = odd \text{ and } i \leq u\}$ and $n_{odd,u}$ be defined in a similar way. Let also α_u be the difference $|n_{even,u} - n_{odd,u}|$, and denote the maximum of α_u for $u=1,\cdots,m$ by α_{max}.

The lists L^i, $i=1,\ldots,2\sigma$ are constructed in the algorithm of Section 2 incrementally by considering the levels $1,2,\cdots$ in order. Let L^i_u be the initial part of L^i which contains those nodes that are in levels $1,\cdots,u$, and let l^i_u be the number of nodes in L^i_u. Clearly, $l^i_m = |L^i|$. Finally, let v_{max} be the maximum number of nodes in a level, and v^*_{max} be the maximum number of nodes in a set of independent nodes in any level. More precisely,

$v_{\max} = \max\{|V_i| ; i=1, \cdots, m\}$ and $v^*_{\max} = \max\{|V_i^k| ; i=1,\ldots,m \text{ and } k=1,\ldots,\sigma\}$.

With this notation, equs (2) may be written in a more general form. More specifically, for any u,

$$|l_u^i - l_u^j| \leq v^*_{\max} \qquad IF\ (i,j \leq \sigma)\ OR\ (i,j > \sigma) \tag{3}$$

But, $\sum_{i=1}^{\sigma} l_u^i = n_{odd,u}$, and $\sum_{i=\sigma+1}^{2\sigma} l_u^i = n_{even,u}$. This, together with (3) gives

$$\frac{n_{odd,u} - (\sigma-1)\, v^*_{\max}}{\sigma} \leq l_u^i \leq \frac{n_{odd,u} + (\sigma-1)\, v^*_{\max}}{\sigma} \qquad \text{if } i \leq \sigma \tag{4.a}$$

$$\frac{n_{even,u} - (\sigma-1)\, v^*_{\max}}{\sigma} \leq l_u^i \leq \frac{n_{even,u} + (\sigma-1)\, v^*_{\max}}{\sigma} \qquad \text{if } i > \sigma \tag{4.b}$$

The above two inequalities, may then be used to compare the number of nodes in two lists L_u^i and L_u^j for $i \leq \sigma$ and $j > \sigma$. Namely,

$$|l_u^i - l_u^j| \leq \frac{|n_{even,u} - n_{odd,u}| + 2(\sigma-1)\, v^*_{\max}}{\sigma} \qquad \text{if } i \leq \sigma \text{ and } j > \sigma \tag{5}$$

From (3) and (5), the following bound may be obtained for any i and j,

$$|l_u^i - l_u^j| \leq \frac{\alpha_u + 2(\sigma-1)\, v^*_{\max}}{\sigma} \qquad \text{for } i,j \leq 2\sigma \tag{6}$$

In order to apply (1), we first need to estimate a lower bound on l_m^i. Noting that $n_{odd,m} + n_{even,m} = n$, the total number of nodes in G_A, and applying the definition of $\alpha_{\max}$, we may obtain,

$$n_{even,m} \geq \frac{n - \alpha_{\max}}{2} \qquad \text{and} \qquad n_{odd,m} \geq \frac{n - \alpha_{\max}}{2}$$

which, together with (4) gives

$$l_m^i \geq \frac{n - \alpha_{\max} - 2(\sigma-1)\, v^*_{\max}}{2\sigma} \qquad \text{for } i = 1, \cdots, 2\sigma \tag{7}$$

Next, we need to estimate an upper bound on the value of ϕ which appears in (1). For this, we consider any two nodes e_1 and e_2 from two different lists L^i and L^j, respectively, such that e_1 and e_2 are neighbors. Being neighbors, the two nodes should either be in the

same level, or be on two consecutive levels. This means that there exists a level u such that e_1 and e_2 are not in L_u^i and L_u^j, respectively, but are in L_{u+2}^i and L_{u+2}^j, respectively. However, L_{u+2}^i may contain at most v_{max}^* nodes which are not in L_u^i, and the same applies to L_{u+2}^j and L_u^j. Hence, from the definition of ϕ we get

$$\phi \leqslant |l_u^i - l_u^j| + v_{max}^*$$

which from (6) gives

$$\phi \leqslant \frac{\alpha_{max} + (3\sigma-2)\, v_{max}^*}{\sigma} \tag{8}$$

Finally, by substituting (7) and (8) into (1) and noting that $2 \leqslant \sigma \leqslant 3$ and that $v_{max} \geqslant 2v_{max}^*$ (guaranteed by the algorithm for coloring the levels), we may obtain the following bound for Δ

$$\Delta \geqslant \frac{n}{2\sigma} - \frac{3}{4}(\alpha_{max} + 2v_{max}) \tag{9}$$

Experimental results, including those for the test problems of Section 6, show that v_{max} is of the order of $\sqrt{n}$. Hence, for large values of n, v_{max} is much smaller than n. Moreover, if the number of nodes in the odd and even numbered levels are approximately equal, then α_{max} is also smaller than n, and (9) may be written in the form

$$\Delta \geqslant \frac{n}{2\sigma} - K \geqslant \frac{n}{6} - K$$

where K is a small number.

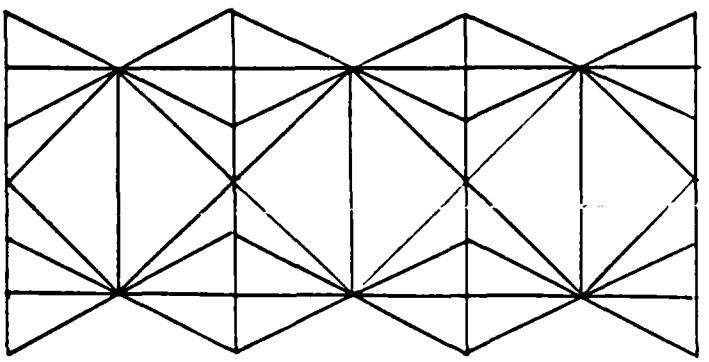


Fig 5 - a mesh resulting in $\alpha_{max} \geqslant \frac{(d-5)\,n}{d-1}$

Although, for the meshes considered in Section 6, α_{max} is much smaller than n, it seems that, without additional restrictions on the topology of the mesh, the value of α_{max} may be of the order of n. For example, a mesh of the type shown in Fig. 5 results in $\alpha_{max} \geq \frac{(f-1)n}{f+1}$, where $f = (d-1)/2 - 1$ and d is the maximum number of neighbors per node (d =9 in the figure). For this type of meshes, the odd/even grouping of the levels leaves the value of α_{max} without any control. In the next section, we discuss a grouping approach which overcomes this problem.

4. A THREE LEVEL GROUPING MECHANISM

The central concept in the algorithm of Section 2 is to group the levels in G_A into two sets of levels, namely $S_{even,m}$ and $S_{odd,m}$, such that the levels in each set are independent. In this section, we replace the odd/even grouping of the levels by another grouping which tries to balance the number of nodes in each group. The flexibility required to maintain this balance may be provided if three, rather than two, groups are used as explained thereon.

Given the levels $V_1, \cdots, V_m$, it is required to distribute these levels into three sets, namely $S_{1,m}$, $S_{2,m}$ and $S_{3,m}$, such that:

1) Any two consecutive levels V_u and V_{u+1} should be in two different sets. This guarantees that the levels in each set are independent, and

2) For $i, j = 1,2,3$, $n_{i,m} \approx n_{j,m}$, where $n_{i,m}$ is the total number of nodes in the levels that are contained in $S_{i,m}$. More precisely, $n_{i,m} = \sum_{V_u \in S_{i,m}} |V_u|$

The sets $S_{i,m}$, $i = 1,2,3$ may be constructed inductively. More specifically, let $S_{i,u-1}$, $i = 1,2,3$ be the sets that were constructed from the levels $V_1, \cdots, V_{u-1}$, and assume, without loss of generality, that $n_{1,u-1} \leq n_{2,u-1} \leq n_{3,u-1}$. Then, in order to keep the size of the sets balanced, it is natural to add level V_u to the set that contains the least number of nodes. In other words:

$$S_{1,u} = S_{1,u-1} \bigcup \{V_u\} \; ; \; S_{2,u} = S_{2,u-1} \text{ and } S_{3,u} = S_{3,u-1}$$

Now if the sizes of $S_{i,u-1}$, $i = 1,2,3$ differ by at most $v_{max} = \max_{1 \leq i \leq m} \{|V_i|\}$, that is if

$$n_{2,u-1} - n_{1,u-1} \leq n_{3,u-1} - n_{1,u-1} \leq v_{max}$$

then it may be shown that after the addition of level u, the sizes of the new sets may not differ by more than v_{max}, that is

$$|n_{i,u} - n_{j,u}| \leq v_{max} \qquad i, j = 1,2,3 \tag{10}$$

Unfortunately, if the set $S_{1,u-1}$ contains level V_{u-1}, then it is not possible to add V_u to that set, because of the dependency between V_u and V_{u-1}. In this case, V_u may be added to

the next smallest set, namely $S_{2,u-1}$. However, the resulting sets may, then, violate (10). More specifically, if $n_{2,u-1} - n_{1,u-1} = D \leq v_{max}$, then it is possible to prove the following least upper bound:

$$n_{2,u} - n_{1,u} \leq v_{max} + D \leq 2v_{max}$$

In order to reduce the maximum difference in the sizes of the sets back to v_{max}, let w be the smallest integer such that one of the following is true

$$|V_{u+1}| + |V_{u+3}| + \cdots + |V_{u+w}| \geq D \qquad (Here\ w\ is\ odd)$$

or

$$|V_{u+2}| + |V_{u+4}| + \cdots + |V_{u+w}| \geq D \qquad (Here\ w\ is\ even)$$

If w is odd, then add $V_{u+1}, V_{u+3}, \cdots, V_{u+w}$ to $S_{1,u}$ and $V_{u+2}, V_{u+4}, \cdots, V_{u+w-1}$ to $S_{3,u}$, while if w is even, add $V_{u+2}, V_{u+4}, \cdots, V_{u+w}$ to $S_{1,u}$ and $V_{u+1}, V_{u+3}, \cdots, V_{u+w-1}$ to $S_{3,u}$. With this, it is easy to show that at level $u+w$, the sizes of the sets $S_{i,u+w}$ satisfy

$$|n_{i,u+w} - n_{j,u+w}| \leq v_{max} \qquad i,j=1,2,3 \qquad (11.a)$$

and that at any level $u+l$ between u and $u+w$ the sizes satisfy

$$|n_{i,u+l} - n_{j,u+l}| \leq 2v_{max} \qquad i,j=1,2,3 \qquad (11.b)$$

After restoring the difference in the sizes of the sets to v_{max}, these sets may be reordered (renamed) such that $n_{1,u+w} \leq n_{2,u+w} \leq n_{3,u+w}$, and the above procedure may be repeated. More precisely, the algorithm may be described as follows:

/* $n_{i,u}$ and t_c will be used to denote the total number of nodes in the levels contained in $S_{i,u}$ and T_c, respectively */

FOR $i=1,2,3$ let $S_{i,0}$ = the empty set

$u = 0$;

REPEAT

$u = u + 1$;

Sort and rename $S_{i,u-1}$, $i=1,2,3$, such that $n_{1,u-1} \leq n_{2,u-1} \leq n_{3,u-1}$;

IF ($V_{u-1} \notin S_{i,u-1}$) THEN

$S_{1,u} = S_{1,u-1} \bigcup \{V_u\}$; $S_{2,u} = S_{2,u-1}$; $S_{3,u} = S_{3,u-1}$;

ELSE {

Let $D = n_{2,u-1} - n_{1,u-1}$.

Let $w = 0$, $c = 1$ and $T_c = T_{1-c}$ = the empty set,

WHILE ($t_c \leqslant D$ and $u + w \leqslant m$) DO

{ $w = w + 1$

$c = 1 - c$ /* switch between T_0 and T_1 */

$T_c = T_c \bigcup \{V_{u+w}\}$ }

$S_{2,u+w} = S_{2,u-1} \bigcup \{V_u\}$;

$S_{1,u+w} = S_{1,u-1} \bigcup T_c$; $S_{3,u+w} = S_{3,u-1} \bigcup T_{1-c}$;

$u = u + w$ }

UNTIL $u \geqslant m$ (the number of levels)

The performance of the algorithm is established in by the following proposition which may be proved from (10) and (11):

Proposition: At any specific stage u of the above algorithm, the number of nodes in the sets $S_{i,u}$, $i = 1,2,3$, may differ by at most $2\, v_{max}$, where v_{max} is the size of the largest level.

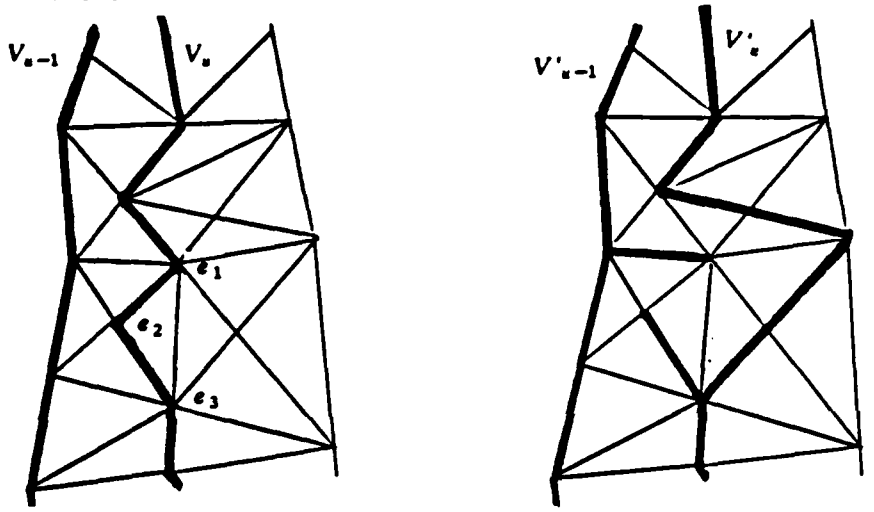
Given that the levels may be partitioned into three sets of independent levels, and that each level may be partitioned into σ sets ($\sigma \leqslant 3$) of independent nodes, then, it is possible to modify the algorithm of Section 2 such that 3σ lists of independent nodes, rather than 2σ lists, are constructed in step 4. The advantage gained from increasing the number of lists is the one of having lists of approximately equal sizes. More precisely, following the same reasoning as in Section 3, it is possible to show that, $n_{i,m} \geqslant (n - 4\, v_{max})/3$ and that $l_m^i \geqslant (n - 4v_{max} - 3(\sigma-1)v^*_{max})/3\sigma$. This leads to the following bound on Δ

$$\Delta \geqslant \frac{n}{3\sigma} - 3.17\, v_{max} \qquad (12)$$

which is better than (9) if $\alpha_{max} \geqslant 2n/9\sigma + 2.12 v_{max}$.

5. ENHANCEMENT OF PERFORMANCE USING BACKTRACKING

Both bounds (9) and (12) depend on the number of colors, σ which are needed to color the nodes of the levels in G_A. Specifically, larger values of Δ are obtained for smaller values of σ. Although, it was proved that $\sigma \leq 3$, experimental results showed that very frequently, two colors are enough for the partition of each level, and that in only few instances, the third color is needed. The need for the third color results whenever a level contains three nodes that are mutual neighbors (see Fig 6(a)). Such nodes clearly form a cycle within the level.



(a) before backtracking (b) after backtracking

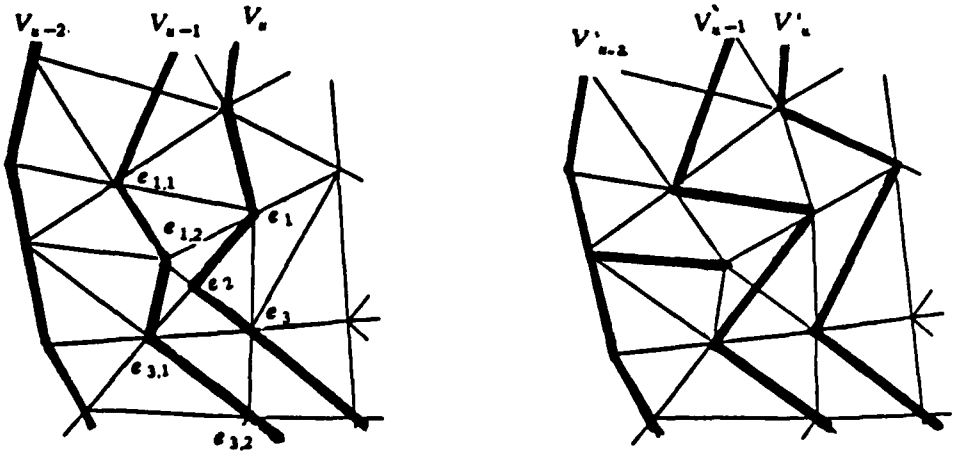
Fig 6 - Example of the one-level look-back scheme

By a slight modification in the level structure of G_A, it is sometimes possible to eliminate the need for a third color. More specifically, assume that $V_1, \cdots, V_{u-1}$ are 2-colorable and that three nodes e_1, e_2 and e_3 form a cycle in level V_u. If one of these nodes, for example e_1, has only one neighbor in the previous level V_{u-1}, then the cycle involving the three nodes may be broken by moving e_1 from level V_u to level V_{u-1} (see Fig 6(b)). The new level $V'_{u-1} = V_{u-1} \bigcup \{e_1\}$ is 2-colorable because V_{u-1} is 2-colorable and e_1 is connected to only one node in V_{u-1}. The following levels, namely V'_w, $w \geq u$, may then be constructed from V'_u in the usual way.

The new level V'_u contains, in addition to all the nodes in $V_u - \{e_1\}$, the neighboring nodes of e_1 which are not in V_{u-1} or V_u. Hence, it is possible for V'_u to contain a cycle.

and thus be 3-colorable. If this is the case, then the above backtracking scheme may be repeated. Note that repeated backtracking may not lead to oscillation because in each instance of backtracking a new node is added to level $u-1$. This may continue until, either level u is 2-colorable or until a cycle is encountered for which backtracking is not possible. In other words, it is possible to construct a level structure in which each level is 2-colorable provided that, whenever a cycle is encountered in some level, one of the nodes forming the cycle is connected to only one node in the previous level.

The above one level look-back strategy may be extented to a two levels look-back scheme. More specifically, assume that three nodes e_i, $i=1,2,3$, form a cycle at level V_u, and that each node e_i, is connected to at least two nodes in V_{u-1}, say $e_{i,k}$, $k \geq 2$ (see Fig 7(a)). In this situation, the one level look-back fails. However, if one of the nodes $e_{i,k}$ is connected to only one node in level V_{u-2}, then we may add this node to V_{u-2} and form the new levels V'_w, $w \geq u-2$. (see Fig 7(b)).



(a) before backtracking (b) after backtracking

Fig 7 - Example of the two-level look-back scheme

The two level look-back scheme should be applied very cautiously because it may lead to execution time which is exponential in the number of nodes. More precisely, assume that, in order to render V_u 2-colorable, a backtrack to level $u-2$ is needed. Level $u-1$ may, then, become 3-colorable, and backtrack to level $u-3$ may be needed. This procedure may roll back all the way to level 1. In order to guard against this, some limit, b_{max}, may

be put on the number of levels allowed to be crossed backward. For example, $b_{max} = 2$ would mean that, if as a result of backtracking from level u, level $u-1$ becomes 3-colorable and backtracking to level $u-3$ is needed, then the backtracking process is stopped, and the 3-color partitioning of level $u-1$ is accepted.

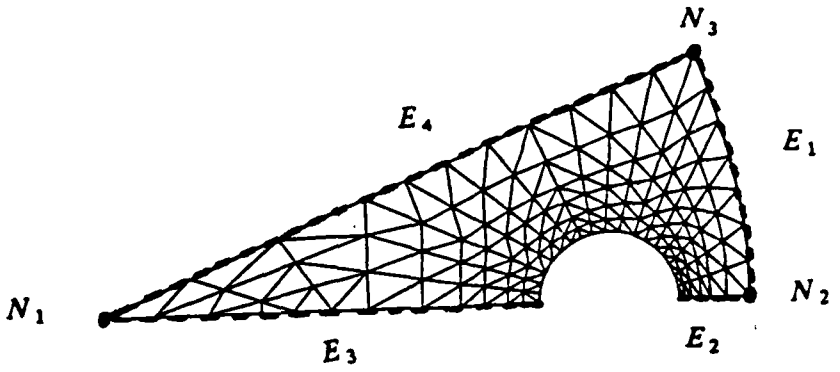
6. EXPERIMENTAL RESULTS

A program which implements the basic numbering algorithm described in Section 2 was written in Fortran. The program incorporates the backtracking schemes of Section 5, and an input variable "BACK" allows for the choice among 0, 1 or 2 level look-back backtracking. Another variable, "Bmax", limits the number of levels that may be rolled back in the case BACK=2. In our experiments, Bmax was fixed at the value 3. A stack oriented approach is used to implement the algorithm of Section 3.2 for coloring the nodes in a particular level.

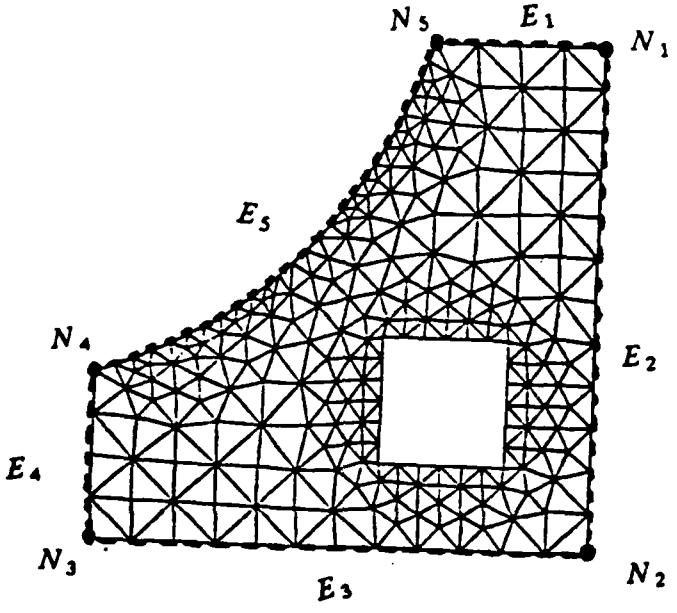
The implementation of the numbering algorithm is general and may be applied to any type of meshes. However, the backtracking facility (BACK > 0) may only be used for triangular meshes. Here we note that, for non-triangular meshes, more than three colors may be needed to color the nodes in each particular level.

In the algorithm of Section 2, 2σ lists of nodes are formed, where $\sigma = \max\{s_u \; ; 1 \leq u \leq m\}$; σ of these lists were formed using the odd numbered levels, and the other σ using the even numbered levels. In implementing the algorithm, a decision was made to distinguish between $\sigma_{even} = \max\{s_u \; ; u = even\}$ and $\sigma_{odd} = \max\{s_u \; ; u = odd\}$. With this, $\sigma_{even} + \sigma_{odd}$ lists of nodes may be formed. Clearly, if $\sigma_{tot} = \sigma_{odd} + \sigma_{even}$, then, $\sigma_{tot} \leq 2\,\sigma$. Although the theoretical bound (9) depends on the largest of σ_{odd} and σ_{even}, it was found that the value of Δ may increase, slightly, due to this optimization.

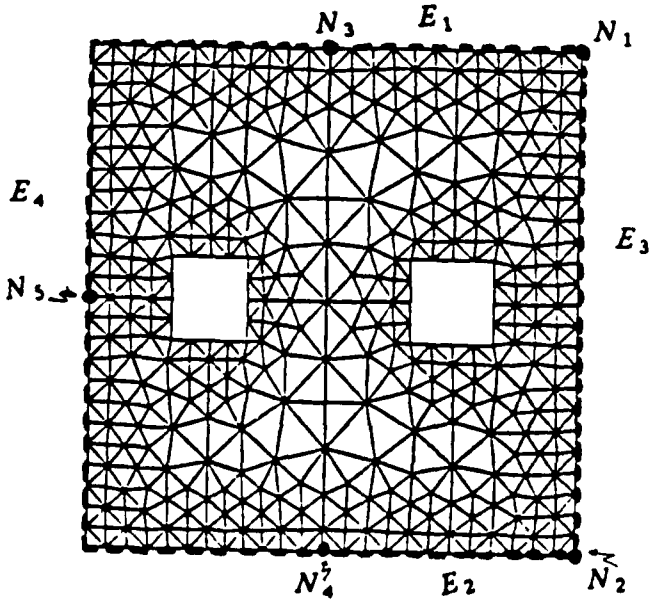
The results of running the algorithm on three test problems generated from the irregular triangular meshes are reported in Tables 1,2 and 3. The test problems are generated from the three meshes $M1$, $M2$ and $M3$ shown in Fig 8. $M1$, which is extracted from [23], contains 145 nodes while $M2$ and $M3$, which are extracted from [9], contain 249 and 449 nodes, respectively. For each mesh, a number of initial levels were tried. An initial level may contain only one node (such a level is denoted by the letter N in Fig 8), or it may contain a number of nodes (such a level is depicted by a dotted bold line in Fig 8 and denoted by the letter E).



(a) The mesh $M1$



(b) The mesh $M2$



(c) The mesh $M3$

Fig 8 - Examples of irregular meshes

In order to evaluate the effectiveness of the backtracking scheme, each problem was tested with BACK set to 0, 1 and 2, respectively. For each run, the total number of lists, $\sigma_{tot} = \sigma_{even} + \sigma_{odd}$, as well as the resulting Δ are reported in the tables. Note that blank entries are used to indicate irrelevant runs. More specifically, if $\sigma_{tot} = 4$ for some value of BACK, then larger values of BACK may not improve Δ.

Initial	BACK = 0		BACK = 1	
Level	σ_{tot}	Δ	σ_{tot}	Δ
N_1	4	33	-	-
N_2	4	33	-	-
N_3	4	32	-	-
E_1	4	31	-	-
E_2	4	33	-	-
E_3	4	30	-	-
E_4	5	20	4	31

Table 1 - The results for the mesh $M1$ (n =145).

Initial	BACK = 0		BACK = 1		BACK = 2	
Level	σ_{tot}	Δ	σ_{tot}	Δ	σ_{tot}	Δ
N_1	6	31	5	37	5	33
N_2	6	26	5	32	4	51
N_3	6	32	5	33	4	51
N_4	6	34	5	32	4	52
N_5	6	33	5	35	4	50
E_1	6	32	5	35	4	51
E_2	6	26	4	47	-	-
E_3	6	31	4	54	-	-
E_4	6	32	5	34	4	53
E_5	6	29	5	33	4	52

Table 2 - The results for the mesh $M2$ (n =249).

The first observation to be made about the results concerns the frequency of success of the backtracking scheme. Among the 26 runs reported, the scheme failed to obtain 2-colorable levels only twice. Namely with $M2$ starting at N_1 and with $M3$ starting at E_4. Also, for a specific problem, larger values of Δ are obtained for smaller values of σ_{tot}, irrespective of the initial level. For example, for $M3$, Δ lies in the range [50,63] if σ_{tot} =6, and in the range [81,99] if σ_{tot} =4. This indicates that the algorithm is not very sensitive to the initial level as long as backtracking is used to minimize σ_{tot}.

Initial	BACK = 0		BACK = 1		BACK = 2	
Level	σ_{tot}	Δ	σ_{tot}	Δ	σ_{tot}	Δ
N_1	6	61	6	54	4	96
N_2	6	63	6	57	4	99
N_3	6	59	6	53	4	94
N_4	6	52	6	50	4	90
N_5	6	56	6	56	4	81
E_1	6	50	6	58	4	93
E_2	6	60	6	52	4	91
E_3	6	56	6	53	4	81
E_4	6	57	6	52	5	58

Table 3 - The results for the mesh $M3$ (n =449).

An interesting phenomena may be observed in Table 3. More specifically, given the symmetry of $M3$, runs with starting levels E_1 and E_2 would be expected to yield the same results. So are runs starting from E_3 and E_4, and runs starting from N_1 and N_2. This is not the case, however, because some initial numbering has to be used to input the mesh into the program, and even though the mesh is symmetric, the initial numbering may not guarantee that the nodes in symmetric triangles are handled in the same relative order. The relative order of nodes is particularly important when backtracking is used because, if more than one node may be used to break a cycle in a level, then the first one encountered is used.

The results also show some expected anomalies. More specifically, a Δ obtained with BACK=k may be worse than the one obtained with BACK=$k-1$. For example, $M3$ starting from N_1 produced Δ = 61 and 54 for BACK = 0 and 1, respectively. This happens because backtracking tends to produce larger levels, thus increasing v_{max}. Hence, if the backtracking procedure fails to reduce σ_{tot}, then, by (9), larger v_{max} leads to smaller Δ.

Although finding the best Δ, namely Δ_{opt} is an NP-hard problem, it is easy to see that for any triangular mesh with n nodes, $\Delta_{opt} \leqslant \frac{n}{3}$. In other words, the best Δ for $M1$, $M2$ and $M3$ may not exceed 48, 83 and 147, respectively. Clearly, our algorithm produces $\Delta > 0.5\Delta_{opt}$ in linear time.

7. CONCLUDING REMARKS

The Multicolor algorithm given in Section 2 is a general numbering algorithm which may be applied to any graph corresponding to an irregular mesh. However, the zero stretch resulting from the algorithm is inversely proportional to the number of colors σ needed to color each level in the graph. For 3-node triangular meshes, it is proved that $\sigma \leqslant 3$, and by applying the same technique used in Section 3, it is possible to prove that, for 4-node rectangular meshes $\sigma \leqslant 4$. Similar tight upper bounds on σ do not seem easy to establish if the nodes are not confined to the corner of the elements and if the number of nodes in each element is not fixed. These types of meshes arise if high order elements are used or if some adaptive mesh refinement techniques are applied [22]. Further research is required to deal with such meshes.

It has been reported [8, 15, 17] that, for linear systems resulting from regular grids, the convergence rate of the PCCG method deteriorates when the zero stretch is increased by renumbering the nodes of the grids (the renumbering also increases the bandwidth). In fact, the results in [15] suggest that the convergence rate is inversely proportional to the zero stretch and/or the bandwidth of the matrix. This effect also exists, in a milder form, in systems resulting from pierced rectangular grids [15], which are slightly irregular grids. If future experiments show that the same effect persists for irregular grids, then it will be interesting to study possible modifications to the multicolor algorithm that will maintain a given zero stretch (specified by the architecture of a supercomputer, for example) while minimizing the bandwidth.

References

1. L. Adams and J. Ortega, "A Multi-color SOR Method for Parallel Computation," *Proc. of the 1982 Int. Conf. on Parallel Processing*, pp. 53-56.

2. G. Chartrand and F. Harary, "Planar Permutation Graphs," *Annals of the Institute of H. Poincare, Sect B*, vol. 3, pp. 433-438, 1967.

3. G. Chartrand, D. Geller, and S. Hedetriemi, "Graphs with Forbidden Subgraphs," *J. of Comb. Th., Ser B*, vol. 10, no. 1, pp. 12-41, 1971.

4. E. Cuthill and J. McKee, "Reducing the Bandwidth of Sparse Symmetric Matrices," *Proc. ACM National Conference*, vol. 24, pp. 157-172, 1969.

5. M. Garey, R. Graham, D. Johnson, and D. Knuth, "Complexity Results for Bandwidth Minimization," *Siam J. on Applied Mathematics*, vol. 34, no. 3, pp. 477-495, 1978.

6. N. Gibbs, "A Comparison of Several Bandwidth and Profile Reduction Algorithms," *ACM Trans. on Mathematical Software*, vol. 2, pp. 322-330, 1976.

7. L. Hageman and D. Young, *Applied Iterative Methods*, Academic Press, New York, (1981).

8. D. Kincaid, T. Oppe, and D. Young, "Vector Computations for Sparse Linear Systems," Tech. Report CNA-189, Center for Numerical Analysis, The University of Texas at Austin, 1984.

9. K. Law and J. Fenves, "A Node-Addition Model for Symbolic Factorization," *ACM Trans. on Mathematical Software*, vol. 12, no. 1, pp. 37-50, 1986.

10. J. Leung, O. Vornberger, and J. Witthoff, "On some Variants of the Bandwidth Minimization Problem," *SIAM J. on Computing*, vol. 13, no. 3, pp. 650-667, 1984.

11. T. Manteuffel, "An Incomplete Factorization technique for Positive Definit Linear Systems," *Mathematics of Computation*, vol. 34, no. 150, pp. 473-497, 1980.

12. L. Marro, "A Linear Time Implemetation of Profile Reduction Algorithms for Sparse Matrices," *SIAM J. on Scientific and Statistical Computing*, vol. 7, no. 4, pp. 1212-

1231, 1986.

13. J. Meijerink and H. van der Vorst, "An Iterative Solution Method for Linear Systems of Which the Coefficient Matrix is a Symmetric M-Matrix," *Mathematics of Computation*, vol. 31, no. 137, pp. 148-162, 1977.

14. R. Melhem, "Determination of Stripe Structures for Finite Element Matrices," *SIAM J. on Numerical Analysis*, 1987. To appear

15. R. Melhem, "Toward Efficient Implementations of Preconditioned Conjugate Gradient Methods on Vector Supercomputers," *Int. J. of Supercomputer Applications*, vol. 1, no. 1, pp. 70-98, 1987.

16. C. Papadimitriou, "The NP-completeness of the Bandwidth Minimization Problem," *Computing*, vol. 16, pp. 263-270, 1976.

17. E. Poole and J. Ortega, "Multicolor ICCG Methods for Vector Computers," Applied Math. Report RM-86-06, University of Virginia, 1986

18. R. Rosen, "Matrix Bandwidth Minimization," *Proc. ACM National Conference*, vol. 23, pp. 585-595, 1968.

19. Y. Saad and M. Schultz, "Parallel Implementations of Preconditioned Conjugate Gradient Methods," Tech. Report YALEU/DCS/RR425,, Dept. of Computer Science, Yale University, Oct. 1985.

20. D.T. Tang, "A Class of Planar Graphs Containing Hamilton Circuits," *IBM Res. Note* , 1965. NC 503

21. J. Turner, "On the Probable Performance of Heuristics for Bandwidth Minimization," *SIAM J. on Comput.*, vol. 15, no. 2, pp. 561-580, 1986.

22. P. Zave and W. Rheinboldt, "Design of an Adaptive, Parallel Finite Element System," *ACM Trans on Mathematical Software*, vol. 5, no. 1, pp. 1-17, 1979.

23. O. Zienkiewicz, *The Finite Element Method, third edition*, McGraw Hill, 1979.

END

JAN.

1988

DTIC